Software Engineering Fundamentals

Introduction to UML
Use Cases

Models and Modeling in UML

- A model is an approximation of reality - it lets us deal with complexity we couldn't otherwise handle.
- UML (Unified Modelling Language) is the product of three different modelling approaches developed by Grady Booch, Ivar Jacobson and James Rumbaugh
- Prior to UML the analysis industry had no single consistent model to follow
  - expensive for companies - which to choose?
  - expensive for system analysts - which to train for?
- It has become the de-facto standard for modelling computing (and non computing) problems.

Origin of UML

- UML 2.0 has 13 different models, split into 2 categories
  - Behavioural diagrams
    - Use Case, Activity, Interaction, State Machine
  - Structural diagrams
    - Class, Object....

Use Cases

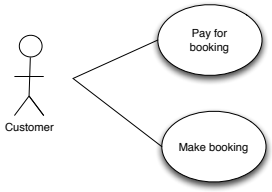
- Many system development failures can be attributed to not involving the client
- The use of Use Cases (UCs) is designed to focus our attention back on what the users are getting out of the system
- Helps us remember to
  - identify all the possible users
  - the different ways they interact with the system (when they initiate a UC)
  - the different ways the system interacts with them (when they are involved in a UC)

Use Cases

- We describe a system in terms of the user's interactions with the system - a Use Case.
- A UC is initiated by an actor (who is external to the system) to accomplish a goal - a *Primary Actor*
  - An person may borrow a book from a library.
- A UC can involve other actors - a *Secondary Actor*
  - The librarian scans the book, marking it as borrowed.
- An actor may play more than one role (set of behaviours) in a UC
  - The insurance assessor may also may also be responsible for liasing with firefighters

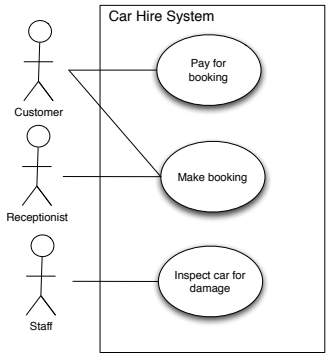
Use Cases

- In this example we assume customers can make bookings via the web - thus they become direct users of the system
- The actors must interact with the system in some way - generally directly.
- Is someone who rings a company a user of the system?
  - What system are we modelling?
  - You need to determine the scope to answer this.



Use Case diagram

- The Use Case Diagram shows the Use Cases in the context of the system (or part of it)
- A use case should be named as an active verb-noun phrase
  - **Make** booking
  - **Inspect** car for damage
  - **Load** container



Why Use Use Cases?

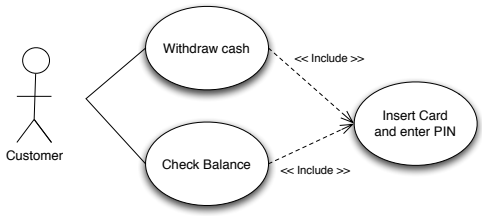
- Good document to work between developers and customers
  - The Use Case Diagram can show the complete functional specification of the system or parts or it
  - Facilitates a "Black Box" approach - customer is only interested in what it does, not how it does it
  - Developer can then take same diagram to next phase to design the system

Use Cases

- A use case is a typical interaction between a user and a computer system.
- A use case captures some function that is visible to the outside world.
- A use case can be small or large
- A use case achieves a discrete goal for the user
- Each use case is a typical interaction between the system being developed and a user or external system
  - A fuller description of a use case includes a step by step list of the actions needed to fulfill its *goal*
- The collection of all use cases is a complete functional description of a system

Use Cases - includes

- Some behaviours are common across use cases
- We can place these into use cases of their own, and connect them via an "includes" association.
- The included use case is never called on its own.

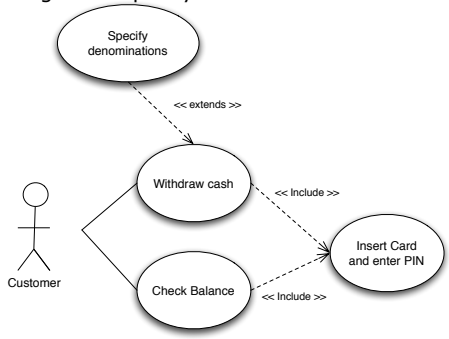


Use Cases - <<extend>> Association

- Many Use Cases will have certain basic behaviour but with any number of variations
- Much clearer than drawing all possible variations as a separate use case is to have further use cases that extend the basic behaviour
- The <<extend>> association allows a basic Use Case to incorporate additional behaviour under certain circumstances.
- <<extend>> is good for modelling
  - Optional parts of Use Cases
  - Complex/Alternative courses of the same interaction
  - Adding new use cases into an existing use case (e.g. more options)

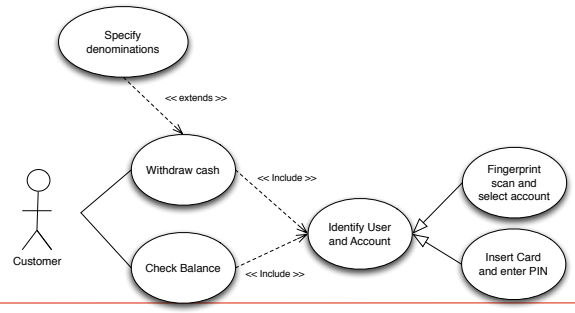
<<extend>> Example

- A withdrawal may normally result in the ATM deciding what notes we get.
- We may specify an extension which allows the user to vary the behaviour by letting them specify the demoninations.



Use Cases - Generalisation

- We can identify use cases that accomplish the same goal but in different ways.
- We can note the similarity by using *Generalisation*
- Both "Fingerprint scan" and "Insert Card" are kinds of "Identify User and Account"



Use Cases - Scenarios

- A use case describes an interaction between a person (or other entity) and a system.
- The purpose is to identify all of the external interactions in a system before progressing onto full analysis of a system.
- The use case diagrams are in a simple form that makes it easy for non technical people to follow
- A use case can be furthered described in English by using a template with fixed fields.
- The use case scenario looks at how an individual use case may occur.

Use Cases - scenarios

- A scenario is an instance of a use case
- It describes one possible set of happenings in a use case.
- The purpose is illustrative
- They help to highlight problems that may exist, but for which we may have no clear solutions at present
- They can be used to test the system when it is finalized

Scenarios

- For example in the context of an ATM system, the Use Case "Validate User" is an interaction between the System and the User.
- Main(primary) Flow of events:
  - System prompts user for a PIN number.
  - User enters PIN number via keypad.
  - User commits PIN by pressing the Enter key.
  - System then checks PIN number to see if it is valid.
  - If the PIN number is valid, system acknowledges the entry.
  - End of Use case.
- Exception Flow of Events
  - System prompts user for a PIN number.
  - User enters PIN number via keypad.
  - User presses Cancel button to cancel transaction.
  - End of Use Case. No changes made to User Account.

Use Case - Text Format

- These allow for much more information about a Use Case
- Structured with...

Use Case Name	
Version	Each revision gets its own version number
Goal	What is supposed to be achieved
Summary	Quick overview of how it works
Actors	
Preconditions	What must be true before this can start
Triggers	An event that causes the use case to start
Basic Course of Events	Normal flow of events - the happy day scenario
Alternative paths	Paths other than the normal flow
Postconditions	The state of the system after the use case
Business Rules	External set of rules a business follows
Notes	Anything extra you may want to record
Author and Date	

How to write Use Cases

- Usually you will develop use cases in discussion with the potential users of a system
- Start with the user goal (for example: "format documents consistently")
- You will need to refine this to come up with the use case from the point of view of system interaction ("define a document style")

Things not in Use Case Diagrams

- Some things can't (and shouldn't) be shown in a Use Case diagram
  - Concurrency
    - each use case transaction is atomic and serialised
  - Individual Objects
    - too low level - interactions should just be between classes
    - E.g. an interaction may be between Person and ATM, but to specify interactions between Bob and the ATM at 42 High St is too detailed
  - Sequence of use cases and interaction between use cases
    - This information is shown elsewhere

Use Cases and Other Diagrams

- As Jacobson described use cases, each use case should eventually result in a sequence diagram
  - This shows how participating objects interact to offer that use case
  - Sequence diagrams in UML were derived from Jacobson's interaction diagrams
  - Sequence diagrams will be explored in detail later